The Choice of the Offspring Population Size in the $(1,\lambda)$ EA

Jonathan E. Rowe
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK

Dirk Sudholt
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK

ABSTRACT

We extend the theory of non-elitist evolutionary algorithms (EAs) by considering the offspring population size in the $(1,\lambda)$ EA. We establish a sharp threshold at $\lambda = \log_{\frac{e}{e-1}} n \approx 5 \log_{10} n$ between exponential and polynomial running times on OneMax. For any smaller value, the $(1,\lambda)$ EA needs exponential time on every function that has only one global optimum. We also consider arbitrary unimodal functions and show that the threshold can shift towards larger offspring population sizes. Finally, we investigate the relationship between the offspring population size and arbitrary mutation rates on OneMax. We get sharp thresholds for λ that decrease with the mutation rate. This illustrates the balance between selection and mutation.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

Keywords

Evolutionary algorithms, comma strategies, offspring populations, runtime analysis, drift analysis, theory

1. INTRODUCTION

Runtime analysis has emerged as a very fruitful research area in the theory of evolutionary algorithms (EAs) and other metaheuristics. By rigorously estimating the expected running time of EAs on interesting classes of problems, we gain valuable insight into the working principles of EAs. Many running time bounds depend on parameters of the algorithm such as the mutation rate or the (offspring) population size. This allows for conclusions about optimal parameters or best possible design choices.

However, most runtime analyses presented so far have been focussing on plus strategies like $(\mu+\lambda)$ EAs, particularly special cases thereof like the $(1+\lambda)$ EA [5, 7] or the $(1+1)$ EA. In these algorithms the new population is chosen among both parents and offspring. On hard problems this bears the risk of getting stuck in local optima. Practitioners therefore often use comma strategies where the new population is selected solely among the offspring.

Analysing non-elitist EAs like these is an interesting and challenging topic as, unlike for plus strategies, one cannot rely on the best fitness in the population to increase monotonically over time. Recent advances include runtime analyses of EAs with ranking selection [10], fitness-proportional selection [12], negative drift results [9] and Lehre's extension of the fitness-level method towards non-elitist EAs [8].

Jägersküpper and Storch [4] were among the first to investigate comma strategies like the $(1,\lambda)$ EA. The $(1,\lambda)$ EA creates λ offspring independently and selects the best among these as parent. Offspring are created by flipping each bit in the parent independently with a given mutation rate u. The default choice is $u = 1/n$, where n is the number of bits.

The choice of the offspring population size λ is crucial. As shown in [4], the algorithm needs exponential time on any function with a unique optimum if $\lambda \leq (\ln n)/14$, with high probability. The reason is that the number of offspring is so small that the algorithm tends to move away from the global optimum once it gets close. This "backward drift" leads to the exponential running time. Contrarily, if $\lambda \geq 3 \ln n$ the algorithm can effectively perform hill climbing and approach the global optimum on easy functions like OneMax. Moreover, there are examples of multimodal functions for which the $(1,\lambda)$ EA is provably more efficient than the $(1+\lambda)$ EA [4].

Note that there is a gap by a large factor of 42 between the upper and lower bounds on λ. This gives little guidance to practitioners as to how precisely λ should be set. We close this gap by showing that there is a sharp phase transition at $\lambda = \log_{\frac{e}{e-1}} n \approx 2.18 \ln n \approx 5 \log_{10} n$ for OneMax. In fact, any smaller value (by a constant factor $1-\varepsilon$) leads to exponential running times on all functions with a unique optimum. Any larger value gives a polynomial expected running time for OneMax. For optimal λ the expected number of function evaluations is $O(n \log n)$. This means that then the $(1,\lambda)$ EA is able to perform simple hill climbing tasks efficiently. In a more general sense, this is a minimum requirement for an algorithm searching for good local optima. Experiments show that this threshold is indeed sharp, if the problem dimension is not too small.

On more complicated unimodal functions the phase transition is shifted to higher λ-values, as demonstrated by theoretical upper bounds, and empirical results. It also depends

GECCO'12, July 7–11, 2012, Philadelphia, USA.

on the mutation rate. When decreasing the mutation rate from its default value $1/n$, the threshold decreases. We make this mutation-selection balance precise and also show that, surprisingly, only two offspring can be effective if the mutation rate is sufficiently small.

Along the way, we also refine popular drift analysis tools. In particular, we investigate the balance between "forward drift" (a tendency to move towards the optimum) and "backward drift" (a tendency to move away from the optimum), in dependence of the offspring population size. This perspective is also useful for analysing parallel EAs; in fact, the (1,λ) EA can be seen as an island model with a complete topology. We are confident that these tools will prove useful in further studies on non-elitist and parallel EAs.

The paper is structured as follows. After some notation has been explained, we revisit and refine drift analysis results such as variable drift for proving upper bounds. We strengthen the negative drift theorem for exponential lower bounds to take account of large self-loop probabilities. We apply these results to determine the exact value for λ for which the run-time on OneMax moves from exponential to polynomial. We extend the polynomial-time result to unimodal functions, and also consider arbitrary mutation rates. Our results are illustrated experimentally, to determine how sharp the thresholds are for realistic problem dimensions.

2. REFINED DRIFT ANALYSIS RESULTS

Drift analysis is a very popular and versatile tool for analysing randomised search heuristics. One typically models the progress throughout the run by a potential function. For the (1,λ) EA a natural potential function is to consider the fitness of the current search point. The fitness function is often taken as a maximisation problem. A trivial transformation leads to the following unified setting where the potential needs to be minimised and the target state is 0.

Throughout this work we define $X_1, X_2, \ldots$ as a sequence of random variables on $0, 1, \ldots, m$, where we think of 0 as the target state. These random variables model the progress of the (1,λ) EA throughout the run, i. e., the index represents the current generation. In all applications we take it as the fitness difference between the current search point and the optimal fitness value. The expected decrease of the current state is called *drift*.

For all $1 \leq k \leq m$, we define:

$$\begin{aligned} p_k^- &= \text{Prob}(X_{t+1} < k \mid X_t = k) \\ p_k^o &= \text{Prob}(X_{t+1} = k \mid X_t = k) \\ p_k^+ &= \text{Prob}(X_{t+1} > k \mid X_t = k) \\ \Delta_k &= E[k - X_{t+1} \mid X_t = k] \\ \Delta_k^+ &= E[X_{t+1} - k \mid X_t = k \text{ and } X_{t+1} > k] \end{aligned}$$

where Δ_k is the drift *towards* the target at state k and Δ_k^+ is the drift *away* from the target, conditional on moving away. For all $0 \leq i, j \leq m$ we abbreviate

$$p_{i,j} = \text{Prob}\left(X_{t+1} = j \mid X_t = i\right).$$

We call the (expected) first hitting time of the state 0 the (expected) *running time* of the process.

2.1 Variable Drift—Nondifferentiable drift

He and Yao [3] presented an additive drift theorem for upper bounds. It gives an upper bound on the expected running time if the drift is at least some positive constant throughout all non-optimal states. The multiplicative drift theorem [2] gives an upper bound when the drift is at least a fixed fraction of the current state. Both drift theorems can be generalised to account for variable drift, where the drift is monotonically increasing with the current state. This was done by Johannsen [6] and, independently, by Mitavskiy, Rowe and Cannings [11]. We present a proof here with slightly weaker assumptions.

Theorem 1. **Variable drift – Johannsen** Suppose there is a monotonic increasing function $h : \mathbb{R}^+ \to \mathbb{R}^+$ such that the function $1/h(x)$ is integrable on $[1, m]$, and with

$$\Delta_k \geq h(k)$$

for all $k \in \{1, \ldots, m\}$. Then the expected running time is at most

$$\frac{1}{h(1)} + \int_1^m \frac{1}{h(x)}\,\mathrm{d}x.$$

Proof. Let

$$g(x) = \begin{cases} \frac{x}{h(1)} & \text{if } x < 1 \\ \frac{1}{h(1)} + \int_1^x \frac{1}{h(z)}\,\mathrm{d}z & \text{if } x \geq 1 \end{cases}.$$

Note that g is strictly monotone increasing and hence invertible. The running time is therefore the same as the first hitting time of the random sequence $g(X_t)$ of the state 0.

If $x \neq 0$ and $y \neq 0$ then

$$g(x) - g(y) = \int_y^x \frac{1}{h(z)}\,\mathrm{d}z \geq \frac{x-y}{h(x)}$$

(since $1/h(z)$ is positive and monotone decreasing) and if $x \neq 0$ and $y = 0$ then

$$g(x) - g(y) = \frac{1}{h(1)} + \int_1^x \frac{1}{h(z)}\,\mathrm{d}z \geq \frac{1}{h(1)} + \frac{x-1}{h(x)} \geq \frac{x-y}{h(x)}.$$

So, for any $k \in \{1, \ldots, m\}$,

$$\begin{aligned} &E[g(X_t) - g(X_{t+1}) \,|\, g(X_t) = g(k)] \\ &= E[g(X_t) - g(X_{t+1}) \,|\, X_t = k] \\ &\geq E[(X_t - X_{t+1})/h(X_t) \,|\, X_t = k] \\ &= \frac{\Delta_k}{h(k)} \geq 1. \end{aligned}$$

So by the classical (additive) drift theorem [3], the first hitting time of 0 by the sequence $g(X_t)$ is bounded above by $g(m)$. The result follows. □

The difference with Johannsen's original version, is that we don't require h to be differentiable. Defining it to be a step function allows us to derive Mitavskiy's version as a corollary:

Theorem 2. **Variable drift – Mitavskiy** Suppose there are $\varepsilon_m \geq \varepsilon_{m-1} \geq \cdots \geq \varepsilon_1 > 0$ such that, for all $0 < k \leq m$

$$\Delta_k \geq \varepsilon_k$$

then the expected running time is at most $\sum_{k=1}^m \frac{1}{\varepsilon_k}$.

Proof. Take

$$h(x) = \varepsilon_{\lceil x \rceil}$$

in Theorem 1. □

2.2 Negative Drift Theorem with Self-Loops

Drift analysis can also be used to prove exponential lower bounds on the running time if there is an additive drift leading away from the target state, on a part of the state space.

The negative (or so-called "simplified") drift theorem of Oliveto and Witt [16], as given in [15], has two assumptions, which we will refer to as **SD1** and **SD2** as follows:

SD1 There exists integers a, b with $0 < a < b \leq m$ and $\varepsilon > 0$ such that $\Delta_k < -\varepsilon$ for all $a \leq k \leq b$. That is, the process drifts backwards (in expectation) between states a and b.

SD2 There exists constants $r, \delta > 0$ (i. e. they are independent of m) such that for all $k \geq 1$ and all $1 \leq d \leq k$

$$p_{k,k-d} \leq \frac{r}{(1+\delta)^d}.$$

That is, there is an exponentially small chance of taking jumps towards the goal state.

The negative drift theorem then states

Theorem 3. **Negative drift**
Suppose a process satisfies conditions **SD1** and **SD2** and T is the running time, starting from $X_1 \geq b$. Let $\ell = b - a$. Then there is a constant $c > 0$ such that

$$\text{Prob}(T \leq 2^{c\ell/r}) = 2^{-\Omega(\ell/r)}.$$

Note that the requirement of having a constant backward drift is quite strong. The negative drift theorem is generally not applicable if the process might spend a significant amount of time performing self-loops. This applies to the (1,λ) EA in many cases if there is a high probability that the fitness of the current search point does not change. One workaround is to replace the random process by another one without self-loops (see [16] for examples).

A more convenient way is to relax the negative drift theorem to account for self-loops. Our new conditions are:

SL1 There exists integers a, b with $0 < a < b \leq m$ and $\varepsilon > 0$ such that $\Delta_k < -\varepsilon(1 - p^o_k)$ for all $a \leq k \leq b$.

SL2 There exists constants $r, \delta > 0$ (i.e. they are independent of m) such that for all $k \geq 1$ and all $1 \leq d \leq k$

$$p_{k,k-d} \leq \frac{r(1-p^o_k)}{(1+\delta)^d}.$$

This gives us the following result:

Theorem 4. Suppose the process satisfies **SL1** and **SL2** and T is the running time, starting from $X_1 \geq b$. Let $\ell = b - a$. Then there is a constant $c > 0$ such that

$$\text{Prob}(T \leq 2^{c\ell/r}) = 2^{-\Omega(\ell/r)}.$$

Note that other variants of the drift theorem [16] can be adapted in the same manner.

Proof. We define a Markov process X^*_t without self-loops, whose transition probabilities are denoted by $p^*_{i,j}$. We show that under the given conditions the original drift theorem can be applied with respect to the starred process. As self-loops only slow down the process, any lower bound for a random first hitting time of the starred process is also a lower bound for the original process.

Let $p^*_{i,i} := 0$ and

$$p^*_{i,j} := \frac{p_{i,j}}{1 - p^o_i}$$

for all $j \neq i$. Also, let $\Delta^*_k = E[k - X^*_{t+1} \mid X^*_t = k]$. Now,

$$\begin{aligned}\Delta_i \leq -\varepsilon(1-p^o_i) &\Leftrightarrow \sum_{j=0}^{m} j \cdot p_{i,j} \leq -\varepsilon \cdot (1-p^o_i) \\ &\Leftrightarrow \sum_{j=0}^{m} j \cdot p^*_{i,j} \leq -\varepsilon \;\Leftrightarrow\; \Delta^*_i \leq -\varepsilon.\end{aligned}$$

This fulfils condition **SD1** for the Markov chain X^*_t. **SD2** is trivially true for $d = 0$, regardless of r and δ. For $d > 0$

$$p_{k,k-d} \leq \frac{r \cdot (1-p^o_k)}{(1+\delta)^d} \;\Leftrightarrow p^*_{k,k-d} \leq \frac{r}{(1+\delta)^d}.$$

Invoking Theorem 3 proves the claim. □

3. FORWARD VS. BACKWARD DRIFT

In a situation, such as the (1,λ) EA, where there can be moves towards and away from the optimum, it is helpful to have the following bound on the overall drift. The following lemma relates the drift to the conditional drift when the algorithm moves away from the optimum.

Lemma 1. Recall that Δ_k is the expected drift towards 0 at k, and that Δ^+_k is the expected drift away from 0, conditional on a backwards move being made. We have

$$\Delta_k \geq p^-_k - p^+_k \Delta^+_k.$$

Proof.

$$\begin{aligned}E[X_{t+1} \mid X_t = k] &= \sum_{j=0}^{k-1} j p_{k,j} + k p^o_k + \sum_{j=k+1}^{m} j p_{k,j} \\ &= \sum_{j=0}^{k-1} j p_{k,j} + k p^o_k + p^+_k(\Delta^+_k + k) \\ &\geq (k-1)p^-_k + k p^o_k + p^+_k(\Delta^+_k + k) \\ &= k - p^-_k + p^+_k \Delta^+_k\end{aligned}$$

and so $\Delta_k \geq p^-_k - p^+_k \Delta^+_k$. □

Notice that when the sequence is elitist, we have $\Delta^+_k = 0$ for all k, and so $\Delta_k \geq p^-_k$. In this situation, the variable drift theorem allows us to recover the well-known method of fitness levels, by defining $\varepsilon_k = p^-_k$ in Mitavskiy's version.

The (1,λ) EA can be regarded as a parallel process where λ mutations are made independently, and the best outcome is taken over for the next population. Even if a single mutation does not give a drift towards the global optimum, the parallel process can—provided the offspring population size λ is large enough. In other words, a sequential process with a drift away from the optimum (if the process has gotten close) can be turned into a parallel process with a drift towards the optimum.

The following result establishes a connection between the process of a single mutation, the offspring population size, and the expected running time of the (1,λ) EA. It will be used in the following sections to establish the phase transition for λ where a backward drift is replaced by a forward drift, if λ increases past the threshold.

Theorem 5. Consider the (1,λ) EA on some finite search space S. Assume a potential function $g\colon S \to \{0,1,\dots,m\}$ so that $g(x^*) = 0$ if and only if x^* is globally optimal for the problem under consideration. For any point x with $g(x) = k > 0$, consider the result, y, of a single mutation. Let

$$p_k^- = \text{Prob}(g(y) < k \mid g(x) = k)$$
$$p_k^+ = \text{Prob}(g(y) > k \mid g(x) = k)$$
$$\Delta_k^+ = \text{E}[g(y) - k \mid g(x) = k \text{ and } g(y) > k].$$

Suppose there exists $\epsilon_m \geq \epsilon_{m-1} \geq \cdots \geq \epsilon_1 > 0$ such that, for all k

$$1 - (1 - p_k^-)^\lambda - \Delta_k^+ (p_k^+)^\lambda \geq \epsilon_k.$$

Then the expected running time of the (1,λ) EA is bounded above by $\sum_{k=1}^{m} \frac{1}{\epsilon_k}$.

In the remainder, we will take $g(x)$ as the fitness difference between x and a global optimum.

Proof. The probability that at least one of the λ offspring is better (i. e., has a smaller potential) than x is $1-(1-p_k^-)^\lambda$. The probability that all offspring are worse than x is $(p_k^+)^\lambda$. And the expected drift away from the target ($k = 0$) conditional on a move away occurring is less than Δ_k^+. Applying Lemma 1, the drift of one generation of the (1,λ) EA is at least

$$1 - (1 - p_k^-)^\lambda - \Delta_k^+ (p_k^+)^\lambda$$

and the result then follows from Theorem 2. □

4. A SHARP THRESHOLD FOR THE OFFSPRING POPULATION SIZE

Before we proceed to the polynomial upper bound, for sufficiently large λ, we require the following lemmas.

Lemma 2. Let $X_0, X_1, \dots$ be the numbers of zeros in the current search point of the (1,λ) EA with mutation rate u. Then for any $0 \leq k \leq n$

$$\Delta_k^+ \leq \frac{1}{(1-u)^{n-1}}.$$

For $u = 1/n$ the bound simplifies to e.

Proof.

$$\begin{aligned}
&\text{E}(X_{t+1} - X_t \mid X_{t+1} > X_t) \\
&= \sum_{d=1}^{n} d \cdot \text{Prob}(X_{t+1} = X_t + d \mid X_{t+1} > X_t) \\
&= \frac{1}{\text{Prob}(X_{t+1} > X_t)} \cdot \sum_{d=1}^{n} d \cdot \text{Prob}(X_{t+1} = X_t + d).
\end{aligned}$$

$\text{Prob}(X_{t+1} = X_t + d)$ is bounded by the probability that d 1-bits flip. Therefore, the sum is bounded by the expected number of flipping 1-bits, $(n - X_t) \cdot u$. The term $\text{Prob}(X_{t+1} > X_t)$ is bounded from below by the probability of mutation flipping only a single 1-bit. The probability for this event is $(n - X_t) \cdot u \cdot (1-u)^{n-1}$. Together,

$$\begin{aligned}
&\text{E}(X_{t+1} - X_t \mid X_{t+1} > X_t) \\
&\geq \frac{1}{(n - X_t) \cdot u \cdot (1-u)^{n-1}} \cdot (n - X_t) \cdot u \\
&= \frac{1}{(1-u)^{n-1}}.
\end{aligned}$$

□

Lemma 3. For any $0 \leq x \leq 1$, and any $n > 0$

$$(1-x)^n \leq \frac{1}{1+nx}.$$

Proof. From $(1-x) \leq e^{-x}$ it follows that $(1-x)^n \leq e^{-nx}$. And from $e^{nx} \geq 1 + nx$ it follows that

$$(1-x)^n \leq e^{-nx} \leq \frac{1}{1+nx}.$$ □

Now we can prove an upper bound on the expected running time of the (1,λ) EA. We consider the well-known function OneMax $= \sum_{i=1}^{n} x_i$ as it reflects how good an EA is in performing a simple hill climbing task. The following result shows that the (1,λ) EA is effective if the population size is large enough.

Theorem 6. If $\lambda \geq \log_{\frac{e}{e-1}} n \approx 2.18 \ln n$ and $u = 1/n$ then the expected number of generations of the (1,λ) EA on the n-bit OneMax function is $O((n \log n)/\lambda + n)$. The expected number of function evaluations is then $O(n \log n + n\lambda)$.

Both upper bounds match the ones for the (1+λ) EA presented in [7] (we refer to bounds for an equivalent parallel EA with a complete communication topology). In this sense, the (1,λ) EA is as effective as the (1+λ) EA with this choice of λ. Note that for $\lambda = \lceil \log_{\frac{e}{e-1}} n \rceil$ we get an upper bound of $O(n \log n)$ function evaluations. This is best possible for EAs using only mutation [18].

Proof. The states are defined by the number of zeros in the current bit string. Let $X_1, X_2, \dots$ be the process defined by randomly mutating a string. For this process, we have $p_k^- \geq k/(en)$, $p_k^0 \geq (1-1/n)^n$ and $\Delta_k^+ \leq e$ according to Lemma 2. By Theorem 5 we have

$$\begin{aligned}
&1 - (1 - p_k^-)^\lambda - \Delta_k^+ (p_k^+)^\lambda \\
&\geq 1 - \left(1 - \frac{k}{en}\right)^\lambda - e\left(1 - \frac{k}{en} - \left(1 - \frac{1}{n}\right)^n\right)^\lambda
\end{aligned}$$

using Lemma 3 and $1/(en) + (1-1/n)^n \geq 1/e$ for $n \geq 2$ [14]

$$\begin{aligned}
&\geq 1 - \frac{1}{1 + \lambda k/(en)} - e\left(1 - \frac{1}{e}\right)^\lambda \\
&= \frac{\lambda k}{en + \lambda k} - e\left(1 - \frac{1}{e}\right)^\lambda \\
&\geq \frac{\lambda k}{en + \lambda k} - \frac{e}{n} \\
&\geq \frac{\lambda k}{2en + 2\lambda k} =: \varepsilon_k
\end{aligned}$$

where the last inequality holds if n, and hence λ, is large enough. Invoking Theorem 2 yields an upper bound of

$$\sum_{k=1}^{n} \frac{1}{\varepsilon_k} \leq 2n + \frac{2en}{\lambda} \sum_{k=1}^{n} \frac{1}{i} = O\left(\frac{n \log n}{\lambda} + n\right).$$ □

Jägersküpper and Storch [4] have shown that the running time of the (1,λ) EA on OneMax is exponential if $\lambda \leq 1/14 \cdot \ln n$. Neumann, Oliveto and Witt [12] showed an exponential lower bound for a similar algorithm in a related setting if $\lambda \leq 1/4 \cdot \ln n$. Here, we relax on these conditions and show that the threshold from Theorem 6 is tight.

Theorem 7. Consider the (1,λ) EA on any function with a unique global optimum. If $\lambda \leq (1-\varepsilon)\log_{\frac{e}{e-1}} n$ for some $0 < \varepsilon \leq 1$ then its running time is at least $2^{cn^{\varepsilon/2}}$ with probability $1 - 2^{-\Omega(n^{\varepsilon/2})}$, for some constant $c > 0$.

Proof. Based on an idea by Doerr, Johannsen and Winzen [1], Sudholt [18] showed that among all functions with a unique global optimum OneMax is the easiest function. This holds with respect to a broad class of mutation-based EAs, which includes the (1,λ) EA. Witt [19] improved this result towards covering stochastic dominance and arbitrary mutation rates $0 < u \leq 1/2$.

Using [19, Theorem 9] we can focus on the running time of the (1,λ) EA on OneMax; the running time on any other function stochastically dominates the former.

We consider the Markov chain defined by the number of zeros in the current bit string, and let $p_{i,j}$ be the transition probabilities. Our goal is to apply the negative drift theorem with self-loop probabilities, Theorem 4.

The interval is chosen as $[0, \ell]$ with $\ell := n^{\varepsilon/2}$. As $\varepsilon \leq 1$ the probability of starting with less than ℓ zeros is $2^{-\Omega(n)}$. In the following, we assume that this does not happen.

As a first step, we replace the original Markov chain by a simpler one where some transition probabilities $p_{k,j}$ are modified in a pessimistic way. If the current state is $k \leq n^{\varepsilon/2}$, we estimate for $j \in \mathbb{N}$

$$p_{k,k-j} \leq \lambda \cdot \binom{k}{j} \cdot n^{-j} \leq \lambda \left(\frac{k}{n}\right)^j \leq \lambda n^{j\varepsilon/2-j} =: p'_{k,k-j}.$$

For transitions to larger indices we assign the probability mass for jumps from k to $k+j$ with $j \geq 2$ to the transition from k to $k+1$. It is clear that this modification is pessimistic. The probability that all λ offspring will have a larger number of zeros than their parent is given by the product of probabilities for each single mutation. A sufficient condition for the latter is that the mutation does flip at least one out of $n-k$ 1-bits and that is does not flip any 0-bit.

We then have

$$\begin{aligned}
\sum_{j=1}^{n-k} p_{k,k+j} &\geq \left(\left(1-\left(1-\frac{1}{n}\right)^{n-k}\right)\cdot\left(1-\frac{1}{n}\right)^k\right)^\lambda \\
&= \left(\left(1-\frac{1}{n}\right)^k - \left(1-\frac{1}{n}\right)^n\right)^\lambda \\
&\geq \left(1-\frac{1}{e}-\frac{k}{n}\right)^\lambda \\
&\geq \left(1-\frac{1}{e}-n^{\varepsilon/2-1}\right)^\lambda \\
&= \left(1-\frac{1}{e}\right)^\lambda \cdot \left(1-\frac{e}{e-1}\cdot n^{\varepsilon/2-1}\right)^\lambda \\
&\geq n^{\varepsilon-1} \cdot \left(1-\frac{e}{e-1}\cdot \lambda n^{\varepsilon/2-1}\right) \\
&\geq n^{\varepsilon-1}/2 := p_{k,k+1}.
\end{aligned}$$

Finally, $p'_{k,k} := 1 - \sum_{j=1}^{k} p'_{k,k-j} - p'_{k,k+1}$. Note that $p'_{k,k} = 1 - \Theta(n^{\varepsilon-1})$, so we indeed have a very large self-loop probability.

Thus,

$$\begin{aligned}
\Delta_k \leq \sum_{j=1}^{k} j \cdot p_{k,k-j} - p_{k,k+1} &\leq \sum_{j=1}^{\infty} j\lambda n^{j\varepsilon/2-j} - n^{\varepsilon-1}/2 \\
&\leq \lambda \sum_{j=1}^{\infty} j(n^{\varepsilon/2-1})^j - n^{\varepsilon-1}/2 \\
&\leq \lambda \cdot \frac{n^{\varepsilon/2-1}}{(1-n^{\varepsilon/2-1})^2} - n^{\varepsilon-1}/2 \\
&\leq -n^{\varepsilon-1}/4 = -\Omega(1-p_{k,k})
\end{aligned}$$

if n is large enough. This establishes **SL1**. For $j \in \mathbb{N}$

$$\begin{aligned}
p_{k,k-j} \leq \lambda n^{j\varepsilon/2-j} &= \Theta(\lambda n^{-\varepsilon/2}) \cdot n^{(j-1)(\varepsilon/2-1)} \cdot \Theta(n^{\varepsilon-1}) \\
&\leq 2^{-j} \cdot (1-p_{k,k})
\end{aligned}$$

if n is large enough. This establishes **SL2** for $\delta := 1$ and $r(\ell) := 1$. Applying Theorem 4 yields that the running time is at least $2^{cn^{\varepsilon/2}}$ with probability $2^{-\Omega(n^{\varepsilon/2})}$, for some constant $c > 0$. This still holds when considering the probability of $1 - 2^{-\Omega(n)}$ for an initialisation with at least ℓ zeros. □

Let us consider experiments to illustrate how drastic the phase transition is. Note that $\log_{\frac{e}{e-1}} n \approx 5 \log_{10} n$. We considered $n \in \{100, 1000, 10000\}$ and stopped each run after either a global optimum has been reached or a maximum of $100n$ generations has passed. The value $100n$ was derived from preliminary experiments; it allows to distinguish successful runs from unsuccessful runs quite clearly.

Figure 1 shows average running time of the (1,λ) EA in 100 runs for each setting. The curves for various problem dimensions have been scaled to fit in one plot. The plot also shows the threshold value $5 \log_{10} n$ in the center of the scale.

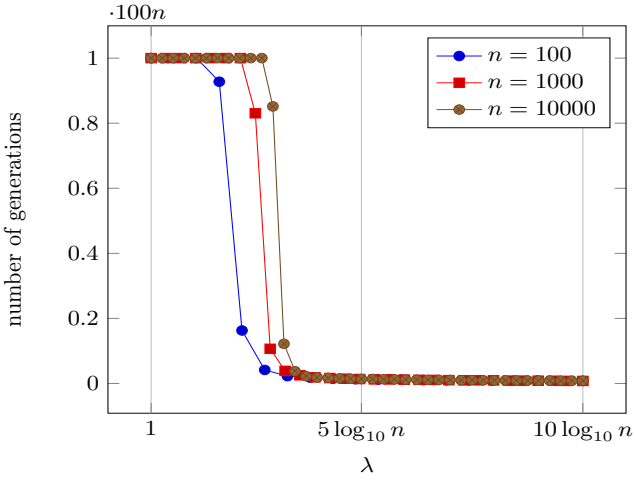

Figure 1: Average number of generations for the (1,λ) EA with mutation rate $1/n$ on OneMax with $n \in \{100, 1000, 10000\}$ bits for all values $\lambda \in \{1, \ldots, 10 \log_{10} n\}$. The number of generations was capped to $100n$. Curves are scaled to fit in one plot.

One can observe that there is a very sharp threshold behaviour. For smaller n the (1,λ) EA seems to be efficient even slightly below the predicted threshold. This does not contradict our lower bound from Theorem 7 as the lower

bound $2^{cn^{\varepsilon/2}}$ is extremely small for small problem dimensions. In accordance with Theorem 7, with growing n the observed threshold value approaches the predicted one.

5. THRESHOLDS FOR UNIMODAL FUNCTIONS

The threshold $\log_{\frac{e}{e-1}} n$ is a lower bound for any function with a unique global optimum. Having an offspring population size above the threshold is sufficient for optimizing OneMax. However, for different problems we might need a larger offspring population size.

We consider the general class of unimodal functions. A function is called unimodal if every non-optimal search point has a Hamming neighbour (i. e., a search point differing in just one bit) with a strictly better fitness.

Theorem 8. Consider an arbitrary unimodal function f with d function values. If $\lambda \geq \log_{\frac{e}{e-1}}(dn) \approx 2.18 \ln(dn)$ then the expected number of generations of the (1,λ) EA on f is $O(dn/\lambda + d)$. The expected number of function evaluations is then $O(dn + d\lambda)$.

Proof. The proof is similar to the proof of Theorem 6. We consider the Markov chain that describes the number of fitness values in between the current fitness and the optimum fitness value. We have $p_k^- \geq 1/(en)$ for all $k > 0$, $p_k^0 = 1 - 1/n$, and clearly $\Delta_k^+ \leq d - k \leq d$. Then repeating previous calculations, we have, for the (1,λ) EA

$$\begin{aligned}
&1 - (1-p_k^-)^\lambda - \Delta_k^+ (1 - p_k^- - p_k^0)^\lambda \\
&\geq 1 - \left(1 - \frac{1}{en}\right)^\lambda - d\left(1 - \frac{1}{en} - \left(1 - \frac{1}{n}\right)^n\right)^\lambda \\
&\geq 1 - \frac{1}{1 + \lambda/(en)} - d\left(1 - \frac{1}{e}\right)^\lambda \\
&= \frac{\lambda}{en + \lambda} - d\left(1 - \frac{1}{e}\right)^\lambda \\
&\geq \frac{\lambda}{en + \lambda} - \frac{1}{n} \\
&\geq \frac{\lambda}{2en + 2\lambda} =: \varepsilon
\end{aligned}$$

where the last inequality holds if n, and hence λ, is large enough. Invoking Theorem 2 yields an upper bound of

$$\sum_{k=1}^{d} \frac{1}{\varepsilon} = \frac{2edn}{\lambda} + 2d. \qquad \square$$

If $d = \Theta(n)$ this doubles the threshold value for λ, compared to the function OneMax, disregarding additive constants.

This additional factor seems to be necessary for some functions. Especially if (unlike for OneMax) there is a risk that the fitness can decrease significantly in a single mutation. The function $\mathrm{LO}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$ (LeadingOnes) counts the number of leading ones in the bit string. An unlucky mutation can destroy a long prefix of leading ones and, if this happens for all offspring, the fitness of the current search point can decrease significantly.

We also consider the function Ridge [17], defined as

$$\textsc{Ridge}(x) := \begin{cases} n - \sum_{i=1}^{n} x_i & \text{if } x \notin \{1^i 0^{n-i} \mid 0 \leq i \leq n\} \\ n + i & \text{if } x \in \{1^i 0^{n-i} \mid 0 \leq i \leq n\} \end{cases}.$$

In a typical run, an EA starts at the bottom of the ridge and then has to walk along the ridge to reach the optimum. If λ is too small, the (1,λ) EA tends to fall off the ridge. Then it typically has to make its way back to the start of the ridge and start over.

Experiments as in Figure 1, for $n = 1000$ and capped at 10^6 generations, show that the phase transition for LO and Ridge is shifted towards larger λ-values. Although LO and OneMax have the same number of fitness values, the former is harder to optimise than the latter, and a larger offspring population size is needed. Also Ridge is harder than LO in this sense.

The thresholds seen in the experiments for LO and particularly Ridge exceed the value $5 \log_{10} n$ (see Figure 2). This justifies why the threshold from Theorem 8 is, in general, worse than the one for OneMax. It indicates that there is a sharp threshold for the offspring population size across various problems, and its location depends on the problem at hand.

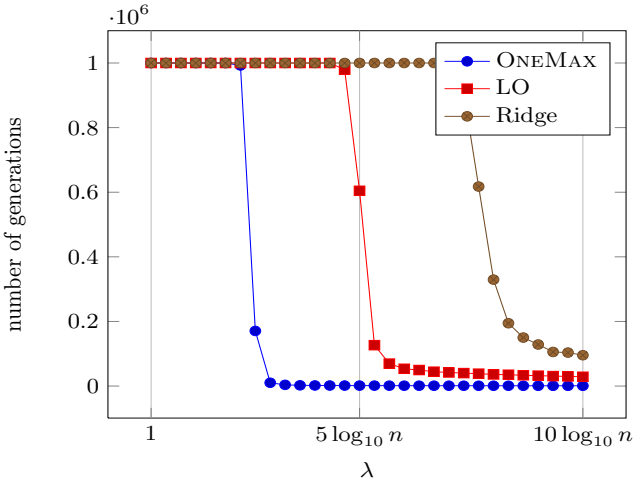

Figure 2: Average number of generations for the (1,λ) EA on OneMax, LeadingOnes, and Ridge with $n = 1000$ bits for all values $\lambda \in \{1, \ldots, 10 \log_{10} n\}$. The number of generations was capped to $n^2 = 10^6$.

6. VARYING THE MUTATION RATE

We now generalise some of our results to arbitrary mutation probabilities $u := u(n)$. The drift depends on u as well as on λ. Similar to work by Lehre and Yao [10] on rank-based selection, we observe a certain mutation-selection balance. That is, the performance of the algorithm depends on the interplay of two parameters. A similar result was also shown in Neumann, Sudholt, and Witt [13] for a comma-strategy variant of an ant colony optimiser. It was shown that $\lambda = 2$ ants result in an expected running time of $O(n \log n)$ on OneMax if the pheromone model adapts quite slowly to new solutions.

The same effect could be achieved by lowering the mutation probability of the (1,λ) EA. For this reason, we generalise our previous bound on OneMax. Here and in the following $H(n) = \sum_{i=1}^{n} 1/i = O(\log n)$ denotes the n-th Harmonic number.

Theorem 9. If

$$\lambda \geq \log_{\frac{1}{1-(1-u)^{n-1}}} \left(\frac{1}{u(1-u)^{2n-2}} \right)$$

then the expected number of generations of the (1,λ) EA with mutation probability u on OneMax with $n \geq 2$ bits is bounded by

$$4n + \frac{4}{\lambda u(1-u)^{n-1}} \cdot H(n).$$

The expected number of function evaluations is bounded by

$$4\lambda n + \frac{4}{u(1-u)^{n-1}} \cdot H(n).$$

Proof. We again consider the Markov chain defined by the number of zeros in the current bit string. Using the notation from Theorem 5 we have $p_k^- \geq (1-(1-u)^k) \cdot (1-u)^{n-k} = (1-u)^{n-k} - (1-u)^n$, the weaker estimate $p_k^- \geq ku \cdot (1-u)^{n-1}$, $p_k^0 \geq (1-u)^n$, and $\Delta_k^+ \leq 1/(1-u)^{n-1}$ for all k by Lemma 2.

Now, we have, using both estimations for p_k^- and Lemma 3,

$$\begin{aligned}
&1 - (1-p_k^-)^\lambda - \frac{1}{(1-u)^{n-1}} \cdot (1 - p_k^- - p_k^0)^\lambda \\
&\geq 1 - (1-ku(1-u)^{n-1})^\lambda - \frac{1}{(1-u)^{n-1}} \cdot (1-(1-u)^{n-k})^\lambda \\
&\geq 1 - \frac{1}{1+\lambda ku(1-u)^{n-1}} - \frac{1}{(1-u)^{n-1}} \cdot (1-(1-u)^{n-k})^\lambda \\
&= \frac{\lambda ku(1-u)^{n-1}}{1+\lambda ku(1-u)^{n-1}} - \frac{1}{(1-u)^{n-1}} \cdot (1-(1-u)^{n-k})^\lambda \\
&\geq \frac{\lambda ku(1-u)^{n-1}}{1+\lambda ku(1-u)^{n-1}} - u(1-u)^{n-1} \\
&= \frac{\lambda ku(1-u)^{n-1} - u(1-u)^{n-1} \cdot (1+\lambda ku(1-u)^{n-1})}{1+\lambda ku(1-u)^{n-1}} \\
&\geq \frac{\lambda ku(1-u)^{n-1} - u(1-u)^{n-1} \cdot (k+\lambda ku(1-u)^{n-1})}{1+\lambda ku(1-u)^{n-1}} \\
&= ku(1-u)^{n-1} \cdot \frac{\lambda(1-u(1-u)^{n-1}) - 1}{1+\lambda ku(1-u)^{n-1}}
\end{aligned}$$

using $u(1-u)^{n-1} \leq u(1-u) \leq 1/4$ and $\lambda \geq 2$

$$\begin{aligned}
&\geq ku(1-u)^{n-1} \cdot \frac{\lambda \cdot 3/4 - 1}{1+\lambda ku(1-u)^{n-1}} \\
&\geq ku(1-u)^{n-1} \cdot \frac{\lambda/3 \cdot 3/4 + 2/3 \cdot \lambda \cdot 3/4 - 1}{1+\lambda ku(1-u)^{n-1}} \\
&\geq ku(1-u)^{n-1} \cdot \frac{\lambda/4}{1+\lambda ku(1-u)^{n-1}} \\
&= \frac{\lambda ku(1-u)^{n-1}}{4+4\lambda ku(1-u)^{n-1}} =: \varepsilon_k.
\end{aligned}$$

Invoking Theorem 5 yields an upper bound of

$$\sum_{k=1}^{n} \frac{1}{\varepsilon_k} \leq 4n + \frac{4}{\lambda u(1-u)^{n-1}} \cdot H(n). \qquad \square$$

For $u = c/n$ with $0 < c \leq 1$, using $(1-c/n)^{n-1} \geq e^{-c}$, we get the following.

Corollary 1. For $u = c/n$ with $0 < c \leq 1$ (possibly depending on n), $\lambda \geq e^c (\ln(n/c) + 2c)$ yields a time bound of $4n + 4ce^c nH(n)/\lambda$ generations, i. e., $4\lambda n + 4ce^c nH(n)$ evaluations.

For larger, but constant c, these calculations are off by a small term as $(1-c/n)^{n-1} \geq e^{-c} - O(1/n)$.

Corollary 2. For $u = c/n$ with $c > 0$ constant, any choice $\lambda \geq e^c (\ln(n/c) + O(1))$ yields an upper time bound of at most $4n + 4ce^c nH(n)/\lambda + O((\log n)/\lambda)$ generations, i. e., $4\lambda n + 4ce^c nH(n) + O(\log n)$ evaluations.

A lesson learnt from these results is that the threshold is very close to $e^c \ln n$, i. e., the minimum feasible offspring population size grows exponentially with the constant c in $u = c/n$.

The threshold for λ from Theorem 9 strictly increases with u. This means that lowering the mutation probability from $u = 1/n$ allows for using smaller offspring populations. It is therefore interesting to ask whether we can do with just $\lambda = 2$ offspring, i. e., we create just two offspring and select the better one. A simple calculation shows that a mutation probability of $u = 1/n^2$ fulfills the condition from Theorem 9.

Corollary 3. If $\lambda = 2$ then $u = 1/n^2$ yields a time bound of $2n^2 \cdot H(n) + O(n \log n)$ generations, i. e., $4n^2 \cdot H(n) + O(n \log n)$ evaluations.

This bound is of order $O(n^2 \log n)$ and hence weaker than the bound $O(n \log n)$ for the ant colony optimiser investigated in [13]. But it shows that polynomial expected running times are possible with just two offspring.

Also the lower bound from Theorem 7 can be generalised, with some additional effort. As the proof is very technical and it uses the same proof ideas as the proof of Theorem 7, it is omitted here.

Theorem 10. Consider the (1,λ) EA with mutation probability $n^{-\delta} < u \leq \varepsilon/4 \cdot (\ln n)/n$, for an arbitrary constant $\delta > 0$, on any function with a unique global optimum. If $\lambda \leq (1-\varepsilon) \log_{\frac{1}{1-(1-u)^n}} \left(\frac{1}{u}\right)$ for some $0 < \varepsilon \leq 1$ then the runtime of the (1,λ) EA is at least $2^{cn^{\varepsilon/3}}$ with probability $1 - 2^{-\Omega(n^{\varepsilon/3})}$, for some constant $c > 0$.

The conditions on the mutation rate imply that λ is always polynomially bounded in n. Apart from the factor $1-\varepsilon$, the threshold for λ differs from the one in Theorem 9 only by the term $1/(1-u)^{2n-2}$ inside the log-term. If $u = \Theta(1/n)$, the two terms only differ by a constant additive term. The difference only becomes important when the mutation rate grows very large.

The balance between the mutation rate and the offspring population size for the OneMax function is illustrated in Figure 3. It shows how the phase transition varies as a function of the mutation rate and the offspring population size λ for $n = 1000$ bits. All runs were stopped after a maximum of 10000 generations.

One can see that here using mutation rate $2/n$ massively shifts the threshold towards higher values. Lower mutation rates lead to a smaller threshold, but at the expense of a higher average running time in cases where λ is large enough.

7. CONCLUSIONS

The analysis of non-elitist EAs is an emerging and interesting topic. We have extended results by Jägersküpper and Storch [4] on the performance of the (1,λ) EA. There is a sharp threshold for the choice of λ at $\log_{\frac{e}{e-1}} n \approx 5 \log_{10} n$.

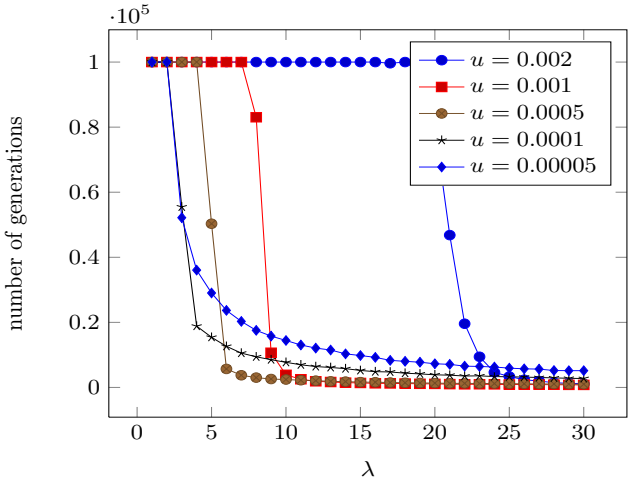

Figure 3: Average number of generations for the (1,λ) EA on OneMax with $n = 1000$ bits, using different mutation probabilities u.

Any smaller values render the algorithm inefficient on larger problems with a unique optimum. Any larger values enable the algorithm to perform hill climbing effectively and hence to optimise OneMax. This is demonstrated by both theoretical and empirical results.

We also conclude that the location of the threshold depends on the problem at hand, as for other unimodal problems the threshold is higher than for OneMax. Finally, there is a delicate mutation-selection balance when varying the mutation rate. Decreasing the mutation rate decreases the minimum offspring population size. With a mutation rate of $1/n^2$ even just two offspring are efficient for OneMax. The results not only apply to the global optimisation of unimodal functions. In a more general sense, they show when the (1,λ) EA is effective at exploitation and can find local optima efficiently.

Acknowledgment

Dirk Sudholt was partially supported through EPSRC grant EP/D052785/1.

8. REFERENCES

[1] B. Doerr, D. Johannsen, and C. Winzen. Drift analysis and linear functions revisited. In *Genetic and Evolutionary Computation Conference (GECCO '10)*, pages 1449–1456. ACM Press, 2010.

[2] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE, 2010.

[3] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

[4] J. Jägersküpper and T. Storch. When the plus strategy outperforms the comma strategy and when not. In *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007*, pages 25–32. IEEE, 2007.

[5] T. Jansen, K. A. De Jong, and I. Wegener. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13:413–440, 2005.

[6] D. Johannsen. *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, 2010.

[7] J. Lässig and D. Sudholt. General scheme for analyzing running times of parallel evolutionary algorithms. In *11th International Conference on Parallel Problem Solving from Nature (PPSN 2010)*, pages 234–243. Springer, 2010.

[8] P. K. Lehre. Fitness-levels for non-elitist populations. In *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, pages 2075–2082. ACM Press, 2011.

[9] P. K. Lehre. Negative drift in populations. In *11th International Conference on Parallel Problem Solving from Nature (PPSN 2010)*, pages 244–253. Springer, 2011.

[10] P. K. Lehre and X. Yao. On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2012. To appear.

[11] B. Mitavskiy, J. Rowe, and C. Cannings. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *International Journal of Intelligent Computing and Cybernetics*, 2(2):243–284, 2009.

[12] F. Neumann, P. S. Oliveto, and C. Witt. Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 835–842, 2009.

[13] F. Neumann, D. Sudholt, and C. Witt. A few ants are enough: ACO with iteration-best update. In *Genetic and Evolutionary Computation Conference (GECCO '10)*, pages 63–70, 2010.

[14] C. P. Niculescu and A. Vernescu. A two-sided estimate of $e^x - (1 + \frac{x}{n})^n$. *Journal of Inequalities in Pure and Applied Mathematics*, 5(3), 2004.

[15] P. Oliveto. *Computational Complexity Analysis of Evolutionary Algorithms for Combinatorial Optimisation*. PhD thesis, Univ. Birmingham, 2009.

[16] P. S. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386, 2011.

[17] R. J. Quick, V. J. Rayward-Smith, and G. D. Smith. Fitness distance correlation and ridge functions. In *Parallel Problem Solving from Nature (PPSN V)*, pages 77–86. Springer, 1998.

[18] D. Sudholt. General lower bounds for the running time of evolutionary algorithms. In *11th International Conference on Parallel Problem Solving from Nature (PPSN 2010)*, pages 124–133. Springer, 2010.

[19] C. Witt. Optimizing Linear Functions with Randomized Search Heuristics - The Robustness of Mutation. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, pages 420–431, 2012.